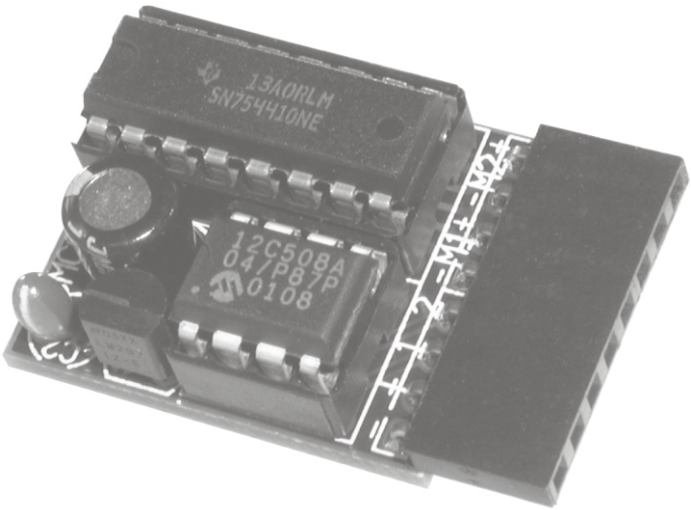
Dual Serial Motor Controller

User's Guide



Contents:

- Safety Warning
- Parts List
- Contacting Pololu
- How to Solder
- Assembly Instructions
- Connecting the Motor Controller
- Using the Motor Controller
- How the Motor Controller Works
- Description and Specifications

http://www.pololu.com/

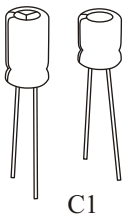
SMC01A

Important Safety Warning

This kit is not intended for young children! Assembly of this kit requires high-temperature soldering and the use of sharp cutting tools. Some included components may become hot, leak, or explode if used improperly. Pololu strongly recommends that you wear safety glasses when building or working with ***any*** electronic equipment. Children should use this kit only under adult supervision. **By using this product, you agree not to hold Pololu liable for any injury or damage related to the use or to the performance of this product. This product is not designed for, and should not be used in, applications where the malfunction of the product could cause injury or damage.**

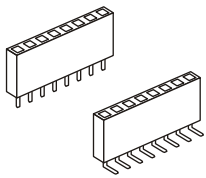
Parts List

The following components are the motor controller parts. Make sure to verify that all components are included, and that you know which component is which. Each component is labeled with its reference number and description. There are 11 parts in the kit, including the PCB.

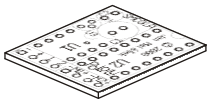


C1
Electrolytic
Capacitor
(2 options)

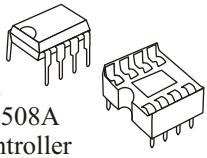
C2
Tantalum
Capacitor



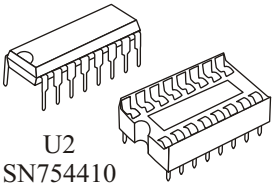
CONNECTOR
8-pin female
header
(2 options)



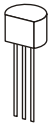
PCB
Printed Circuit
Board



U1
PIC12C508A
Microcontroller
and Socket



U2
SN754410
Dual H-Bridge
and Socket



U3
LM2931
5-volt Regulator

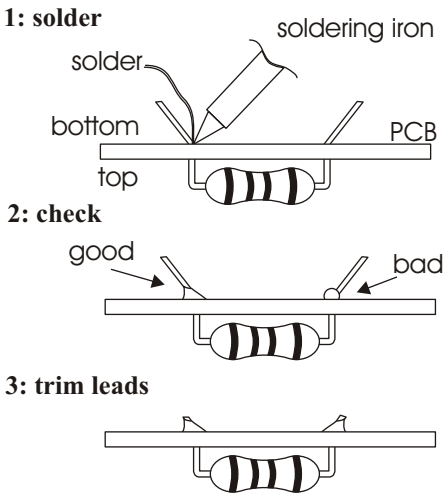
Contacting Pololu

You can check the Pololu web site at `http://www.pololu.com/` for latest information about the motor controller, including color pictures, application examples, and troubleshooting tips.

We would be delighted to hear from you about your project and about your experience with our motor controller. You can contact us through our online feedback form or by email at `support@pololu.com`. Tell us what we did well, what we could improve, what you would like to see in the future, or anything else you would like to say!

How to Solder

You need a soldering iron and diagonal cutters to assemble the motor controller. The green printed circuit board (PCB) is the base that holds the components together and establishes the necessary electrical connections. The PCB has two sides: a top side, or *component side*, which has white silkscreen markings, and a bottom side, or *solder side*. Insert the components from the top side and solder them on the solder side. In general, you should insert and solder the components so that they are as close as possible to the PCB. All components in this kit, except for voltage regulator U3, should be flush with the PCB. After soldering, trim excess leads with diagonal cutters.



To solder, heat a component lead and the PCB pad and then apply solder until the solder flows onto both the lead and the pad. If the solder beads up on the lead or on the pad, the connection is bad, so you should apply more heat. However, be careful not to damage any components through overheating.

Assembly Options

You can assemble the motor controller in several ways, so before you begin, there are three choices you must make. The options concern the size of the assembled motor controller. You can insert the various components into the PCB without soldering to help determine which options are best for you.

Should I use IC sockets for U1 and U2? The sockets are not necessary, and not using them will make your motor controller slightly smaller. Not using them, however, will make it much more difficult to replace the ICs if they are damaged, and also make it possible for you to damage the devices while soldering. **We strongly recommend using the sockets.**

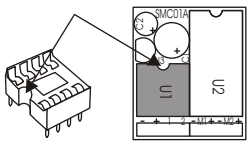
Which C1 option should I use? Two electrolytic capacitors are provided in the kit. The larger capacitor is rated for 25 volts, whereas the smaller one is rated for 16 volts. These ratings limit the voltage you can apply to the motor controller and thus limit the maximum voltage with which you can drive the motors. **If you want to drive the motor with voltages higher than 16 volts, or if you don't care about size, use the larger capacitor.**

Which connector should I use? A straight connector and a right-angle connector are provided in the kit. It is up to you which you choose to use; you can also solder motor leads directly into the PCB holes and avoid using a connector altogether.

Assembly Instructions

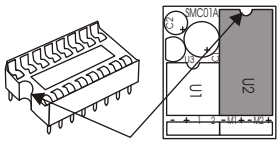
Caution: The components U1-U3 can be damaged by static electricity. Ground yourself (touch a water pipe or the metal frame of a piece of electrical equipment) before handling these components, and avoid touching their leads. Once assembled, the motor controller can be stored safely in the conductive bag in which the kit is packaged.

1



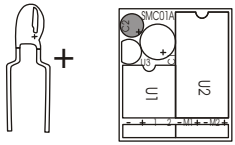
Insert the 8-pin socket from the top side of the PCB in the area indicated U1, and solder. On one side of the socket is a notch that should be aligned with the notch on the PCB drawing. The socket protects the PIC microcontroller from being damaged during soldering.

2



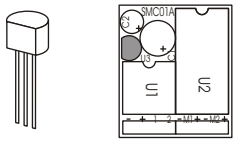
Solder the 16-pin socket in the area indicated U2. Make sure to align the notch on the socket with the notch on the PCB drawing. The socket allows you to replace the motor driver chip if it ever breaks.

3



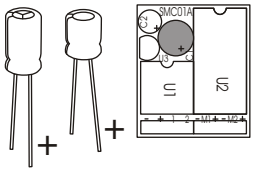
Next, add the tantalum capacitor C2. You may need to bend the leads to make them straight so that they will fit. The capacitor is *polarized*, which means it **must only go in one way**. Make sure the side labeled with a "+" and a stripe goes into the hole that is also marked with a "+". If the PCB is oriented as shown in the diagram, the lower hole is for the positive lead.

4



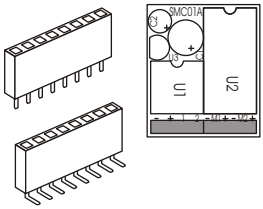
Now, add the voltage regulator, U3. Make sure the device is oriented as shown on the silkscreen drawing, with the flat side facing the outside of the PCB. **Caution! The voltage regulator can be damaged by static electricity.**

5



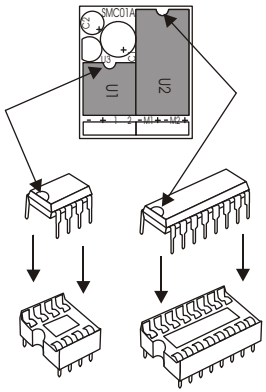
In this step, add your choice of electrolytic capacitor for C1. The larger capacitor is necessary if you want to run your motors off of a 24-volt power source, and you should use it if you don't care about the size of the completed motor controller. **The electrolytic capacitors are polarized, but this time the stripe identifies the negative terminal.** Also, the positive lead is longer. Make sure to match up the positive lead with the appropriate hole in the PCB.

6



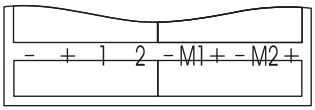
Next, solder in your choice of connector. You can use either the straight or the right-angle female header provided in your kit, or use your own connector. If you prefer having leads soldered directly to the PCB, you can do so now or wait until you have the leads ready and available.

7



Finally, insert the two integrated circuits (ICs), U1 and U2, into their sockets. **Make sure that you plug them in so that the notches on the ICs match the notches on the PCB outline. Do not solder the ICs to the sockets.** If you soldered the sockets in backwards, it doesn't matter as long as the actual component is oriented correctly. **Be careful: the ICs are static-sensitive**, so take appropriate precautions and avoid touching their leads. You may need to bend some of the leads to make them fit in the sockets; if so, hold the plastic body of the IC and push the pins (gently!) against a flat surface, one row at a time.

Connecting the Motor Controller



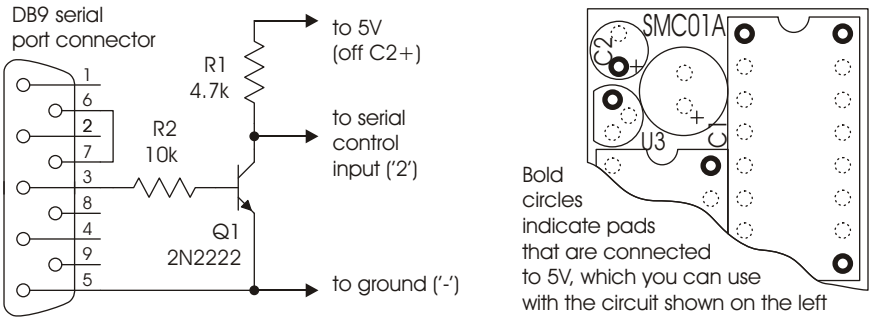
There are eight pins on the bottom of the motor controller for connecting it to the rest of your system. A closeup of the bottom of the PCB is shown to the right, in case you have a hard time reading the silkscreen on your board. The eight pin labels and the corresponding functions are shown in the table.

LABEL	FUNCTION
-	ground (0V)
+	positive supply (5.6-25V)
1	reset
2	serial control input
M1-	motor 1, negative output
M1+	motor 1, positive output
M2-	motor 2, negative output
M2-	motor 2, positive output

Connecting Power. Warning: connecting power incorrectly can cause some components to explode. Connect the ground pin to a ground terminal on your robot controller. If you have a separate power supply for just the motors, make sure that you connect the negative terminal of that supply to the same ground. (This situation may arise if, for example, you want to run your robot controller off of a 9-volt battery and you want to run your motors off of a 12-volt battery. You will also need an independent power supply for the motors if you want to use a personal computer as the robot controller. In that case, you might use a battery for the motor supply and use a wall outlet for the PC supply.) Connect the '+' pin to the positive terminal of the motor supply; this terminal may connect only to the motor controller, or it may connect to any other device powered by that supply. **Warning**: the supply voltage may not exceed 16 volts or 25 volts, depending on which capacitor you chose for C1 in step 5 of assembly.

Reset Input. The reset input is optional, but you may need to use it to ensure that spurious signals sent when your robot controller turns on do not cause the motor controller to detect the baud rate incorrectly. Connect this pin to a 0-5V digital output on your robot controller. The line should normally be kept high (+5V), but bringing it low (to 0V) for at least 2 microseconds resets the motor controller to its initial state (all motors off, waiting for its first serial command).

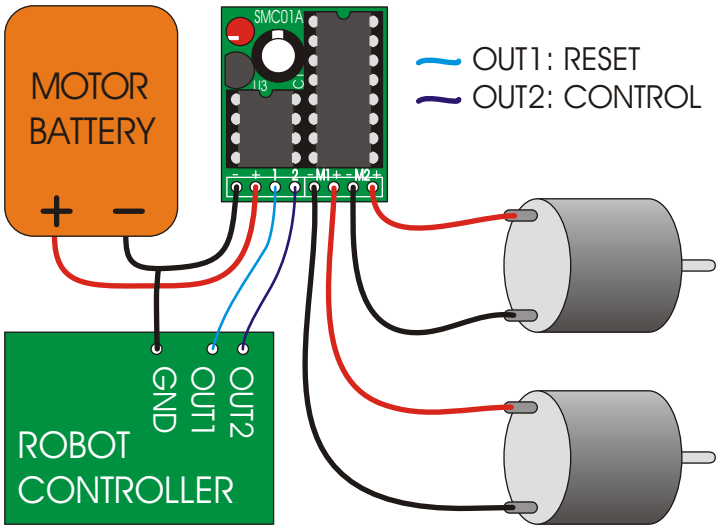
Serial Input. Use a pin on your robot controller that can be used as a TTL-level, asynchronous serial output. Serial data can be sent down this line 8 bits at a time, with no parity bit, at any rate between 1200 and 19200 baud. *Once you choose a baud rate, you cannot change it until the motor controller is reset.* **Important note:** unlike RS-232 serial lines (the standard for serial ports used to connect devices to personal computers), this line uses TTL voltages (between 0 and 5 volts). The higher voltages used on RS-232 lines will damage the motor controller. If you need to convert RS-232 levels to TTL levels, you will need to use a level converter such as the MAX220 (made by Maxim). You could also use the simple circuit shown at the top of the next page. **When building circuits that connect to a PC, be especially careful because you could potentially destroy the PC's serial port. Before attempting to connect your own electronics to a computer, make sure you know what you are doing!**



The above diagram shows a simple circuit for connecting the motor controller to a PC serial port. You will need to connect one side of resistor R1 to a 5V supply, which is available on the PCB at the points indicated on the figure to the right. You can solder a wire onto one of the pads, but make sure that the wire touches only the intended pad.

Connecting the Motors. Connect one or two motors to the pins labeled M1 and M2. You probably don't need to worry too much about the polarity, but the '+' pins go positive when the controller receives a "forward" command. If you find out that your motors turn in different directions than you expect, you can flip the wiring or just switch the forward and reverse commands on your robot controller program.

A typical setup is shown in the diagram below. Keep in mind that the wiring you use for the motor outputs and power connections should be capable of conducting several amps. We recommend using at least 26 gauge wire (remember, smaller numbers mean bigger wires!).

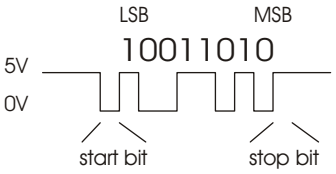


Using the Motor Controller

To set the speed and direction of a motor, send a four-byte command with the following structure to the motor controller's asynchronous serial input, labeled '2' on the PCB.

start byte = 0x80	device type = 0x00	motor # and direction	motor speed

You must send the four-byte command eight bits at a time (with no parity bit) at a constant baud rate ranging from 1200 to 19200 baud. The serial bits must be *non-inverted*, meaning that a zero is sent as a low voltage, and a one is sent as a high voltage, as shown in the diagram to the right. (The PC-connection circuit on the previous page corrects the inverted signal coming out of PC serial ports.) *Commands sent to the serial input* ***must*** *conform to the above format (described in detail below) or else the motor controller and other devices connected to the serial line may behave unexpectedly.* This motor controller *interface protocol* is compatible with other Pololu serial devices such as our servo controller, so you can control multiple Pololu serial devices on a single line.

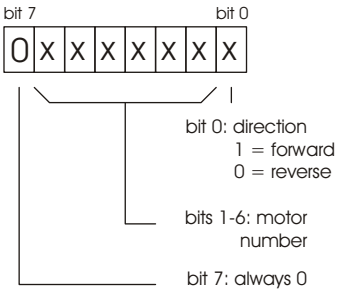


The Four-byte Motor Controller Command

Byte 1: Start Byte. This byte should *always* be 0x80 (128 in decimal) to signify the beginning of a command. The start byte is the only byte with the highest bit (bit 7) set, and it alerts all devices on the serial line that a new command is being issued. All succeeding bytes sent down the serial line must have their highest bit cleared to zero.

Byte 2: Device Type. This byte identifies the device type for which the command is intended, and it should be 0x00 for commands sent to motor controllers. All devices that are not SMC01A dual motor controllers ignore all subsequent bytes until another start byte is sent.

Byte 3: Motor Number and Direction. This byte has three parts, as shown in the diagram to the right:



Bit 0 specifies the direction of the motor. Set this bit to 1 to make the motor go forward; clear the bit to make it go backward.

Bits 1-6 specify the motor number. If you are using only two motors per serial line, you can use the default values of 0 for motor M1 and 1 for motor M2. If you want to control more than two motors, use numbers in the range of 2 to 63, as described in the section, "Controlling Multiple Motor Controllers with One Serial Line".

Byte 3: Motor Number and Direction (continued).

Bit 7 must be cleared since this is not a start byte.

To obtain the complete byte 3 value from a motor number and a direction, multiply the motor number by 2 and add 1 if the direction is forward. For example, to make motor 5 go forward, byte three should be 5 x 2 + 1 = 11. To make motor 1 go backward, byte 3 should be 1 x 2 = 2. (Two efficient ways to multiply by 2 in a microcontroller program are shifting left by one digit or adding the motor number to itself.)

Byte 4: Motor Speed. The most significant bit must be zero since this is not a start byte, and the remaining seven bits are available for specifying the speed. The possible range of values for byte 4 is thus 0x00 to 0x7F (0 to 127 decimal). 0x00 turns the motor off, and 0x7F turns the motor fully on; intermediate values correspond to intermediate speeds.

Resetting the Motor Controller

The motor controller's optional reset line should normally be kept high at +5V. Pull the reset line low to 0V for at least 2 microseconds to reset the motor controller to its initial state (all motors off, waiting for the first serial command). You do not need to reset the motor controller to use it successfully. However, you may need to reset the motor controller to ensure that spurious signals sent when your robot controller turns on do not cause the motor controller to detect the baud rate incorrectly.

Controlling Multiple Motor Controllers with One Serial Line

To control a particular motor, you must specify its motor number in command byte 3. For all motor controller boards, motor M1 responds to commands for motor number 0, and M2 responds to commands for motor number 1. To control more than two motors with a single serial line, you need to use motor numbers 2 through 63. Motor controllers that are not specially ordered respond to numbers 2 and 3; you need to order specially programmed motor controllers to use motor numbers 4 through 63.

For example, to control six motors independently, you need three motor controller boards, each with different motor numbers. All three motor controllers respond to commands for motor numbers 0 and 1. For controlling the six motors independently, use motor numbers 2, 3, 4, 5, 6, and 7. (The exact numbers depend on which motor numbers you request when you specially order additional motor controllers.)

You can individually control up to 62 motors at a time with a single serial line using 31 motor controllers: one with the default program and 30 that are specially programmed.

Example BASIC Stamp II Program

This program, which can run on a BASIC Stamp II controller, makes motor 1 gradually speed up, then slow down, then speed up in the other direction, and then slow down again. For the code to work, pin 15 must be connected to the reset input ('1'), and pin 14 must be connected to the serial input ('2'). The interface code should look similar in other programming languages; the description below should help you in understanding the code and, if necessary, in translating it to other languages.

On line 1, the 8-bit variable `speed` is declared for later use. The motor controller is then reset by a low-going pulse on pin 15 (lines 2 and 3).

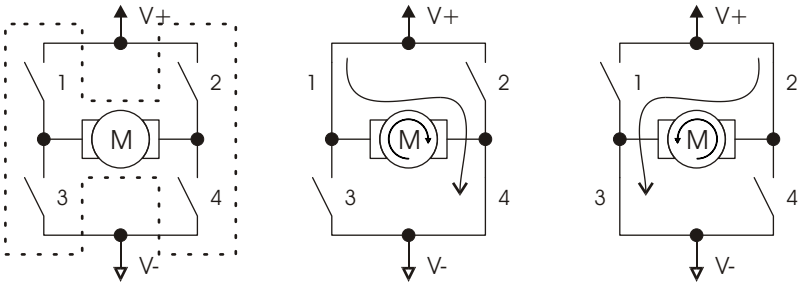
The first *for loop* on lines 4-7 causes motor 1 to gradually speed up. The serial output is created by the `serout` statement on line 5. The first parameter, 14, specifies the pin number through which to send the serial signal. The next parameter, `32`, sets up the serial characteristics to be 8 bits with no parity, non-inverted, at a baud rate of 19200. The four numbers in square brackets are the data to be sent, and they correspond to the four control bytes for the motor controller. The first two bytes should always be `$80` and `0`. The second `0` makes motor 1 go backward. The speed variable, which increases every time through the loop, is the only part of the command that changes, and that is what makes the motor gradually speed up. The `pause` statement on line 6 causes the program to wait for 20 ms (0.02 seconds) before sending the next command.

When the first loop ends, the motor is set to its full speed of 127. The second loop on lines 8-11 slows the motor back down by sending speeds from 127 down to 0. The next two loops on lines12-19 then repeat the process, except for the parameter value of `1` in byte three, which causes motor 1 to spin forward.

```
1       speed   var  byte
2       low 15       'reset motor controller
3       high 15
4       for speed = 0 to 127
5          serout 14,32,[$80, 0, 0,speed]
6          pause 20
7       next
8       for speed = 127 to 0
9          serout 14,32,[$80, 0, 0,speed]
10         pause 20
11      next
12      for speed = 0 to 127
13         serout 14,32,[$80, 0, 1,speed]
14         pause 20
15      next
16      for speed = 127 to 0
17         serout 14,32,[$80, 0, 1,speed]
18         pause 20
19      next
```

How the Motor Controller Works

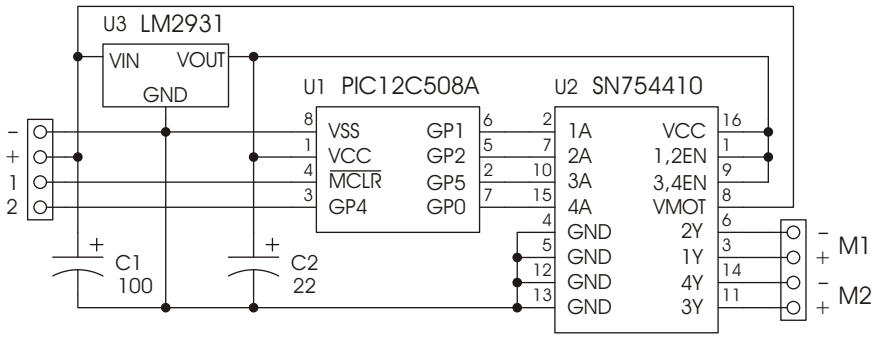
The motor controller uses *H-bridges* to turn motors forward and backward (see the dotted 'H' in the left figure). H-bridges have four switches, which are turned on in pairs to allow current to flow into the motors in both directions, as shown below. In the left figure, all four switches are open, and the motor is turned off. When switches 1 and 4 close, the motor turns in one direction; when switches 2 and 3 close, the motor turns the other way. Integrated circuit U2 contains two H-bridges, allowing bidirectional control of two motors.



A technique called *pulse width modulation* (PWM) is used to control the speed of the motors. The microcontroller (U1) is a little computer that controls the H-bridge switches. It turns the switches on and off very rapidly (600 times per second) and varies the percentage of the time that the switches are on to achieve the speed set by the serial interface. For a higher speed, the switches are on a larger fraction of the time than for a slower speed. At the maximum speed of 127, the switches are left on. The momentum of the motor shaft keeps the shaft spinning at a constant speed that can be varied smoothly over all 127 different speeds.

U3 and the capacitors provide a steady 5V power supply for the microcontroller, which cannot run at the full motor voltage provided to the board at the '+' and '−' pins.

The complete schematic diagram of the motor controller is shown below:



The Pololu Dual Serial Motor Controller

For a robot to interact with its environment, it must be able to convert electrical signals into motion. However, the power requirements of *actuators*, electrical devices capable of producing motion, are typically so high that normal digital circuitry cannot drive them. In addition, precise motion control requires constantly changing the signals sent to the actuators, leaving the control circuitry with little time to attend to other tasks.

The Pololu motor controller bridges the gap between robot controllers and power-hungry actuators. Using one serial output from your robot controller, you can independently set each of two small DC motors (the kind typically found in remote-control cars and motorized toys) to go forward or backward at any of 127 different speeds. To control additional motors, you can connect multiple motor controllers to the same serial line. The motor controller is compatible with the Pololu Servo Controller, so you can control an almost unlimited number of motors and servos with one serial line. Because of its small size, the motor controller can fit almost any robot design.

Specifications

PCB size	1.00" x 0.85"
Motor ports	2
Speeds	127 forward, 127 backward, and off
Maximum current	1 A per motor (continuous)
Supply voltage	5.6-25 V
Data voltage	0 and 5 V
PWM frequency	600 Hz
Serial baud rate	1200-19200 (automatically detected)